Définition Formelle — Glushkovizer

Résumé

Ce document constitue la définition formelle des différents types de donnée utilisés tout au long de cette librairie. Dans un premier temps, nous nous concentrerons sur les expressions régulières et les fonctions définies sur celle-ci. Puis dans un second temps, nous nous intéresseront aux automates et à leur divers fonctions définies sur eux. Enfin pour finir, nous parlerons d'automates particuliers, ceux de Glushkov, nous aborderons leurs constructions, ainsi que leurs propriétés.

I Les expressions régulières

Dans cette section, nous parlerons d'expressions régulières (*ER*). Nous allons nous concentrer sur un type bien particulier d'expressions régulières qui ne seront pas les expressions régulières que nous pouvons voir plus quotidiennement dans le domaine de l'informatique, les expressions régulières *UNIX*. Mais plutôt une version plus simple de celles-ci.

1 Définition

Nous allons noter une expression régulière $E(\Sigma)$, c'est-à-dire une expression régulière où les symboles sont inclus dans l'ensemble Σ. Cette expression reconnait un langage qu'on pourra appeler $L(E(\Sigma))$. Nous pouvons définir une expression régulière récursivement de cette manière :

$$E(\Sigma) = \epsilon \tag{1}$$
$$E(\Sigma) = a \quad \text{avec } a \in \Sigma \tag{2}$$
$$E(\Sigma) = F(\Sigma) + G(\Sigma) \tag{3}$$
$$E(\Sigma) = F(\Sigma) \cdot G(\Sigma) \tag{4}$$
$$E(\Sigma) = F(\Sigma)^* \tag{5}$$
$$E(\Sigma) = (F(\Sigma)) \tag{6}$$

On notera que $*$ est prioritaire sur $\cdot$ qui est lui-même prioritaire sur $+$ et qu'ils sont tous deux associatif à gauche. On comprend donc pourquoi l'équation (6) existe, elle est là pour des raisons de priorité. Il est alors évidant de calculer les diverses fonctions sur celle-ci, c'est pour cela qu'on ne précisera pas son calcul. On peut définir chaque équation comme ceci :

- $E(\Sigma) = \epsilon$ **(epsilon) :** Représente le mot vide, de ce fait un mot de longueur zéro. Il en vient que :

$$L(E(\Sigma)) = \{\epsilon\} \tag{7}$$

  Il peut être parfois représenté par '$'.

- $E(\Sigma) = a$ **:** Représente un symbole présent dans l'ensemble Σ. Il en vient que :

$$L(E(\Sigma)) = \{a\} \tag{8}$$

- $F(\Sigma) + G(\Sigma)$ **:** Représente l'union des deux expressions régulières $F(\Sigma)$ et $G(\Sigma)$. Il en vient que :

$$L(F(\Sigma) + G(\Sigma)) = L(F(\Sigma)) \cup L(G(\Sigma)) \tag{9}$$

Par abus de langage, on peut aussi dire $F(\Sigma)$ 'ou' $G(\Sigma)$ pour représenter cette union.

- $E(\Sigma) = F(\Sigma) \cdot G(\Sigma)$ **:** Représente la concaténation deux deux des deux expressions régulières $F(\Sigma)$ et $G(\Sigma)$. Il en vient que :

$$L(E(\Sigma)) = \{u.v \mid u \in L(F(\Sigma)) \wedge v \in L(G(\Sigma))\} \tag{10}$$

- $E(\Sigma) = F(\Sigma)^*$ **:** Représente la répétition infinie de $F(\Sigma)$, cette répétition incluant, aucune répétition et donc le mot vide. Il en vient que :

$$L(E(\Sigma)) = \{u^* \mid u \in L(F(\Sigma))\} \tag{11}$$

Exemple :

On comprendra ainsi que l'expression $E(\mathbb{A}) = a + c.d$ avec $\mathbb{A} = \{a, b, c, d\}$, dénote le langage $L(E(\mathbb{A})) = \{a, cd\}$. Car on peut représenter $E(\mathbb{A})$ comme ceci :

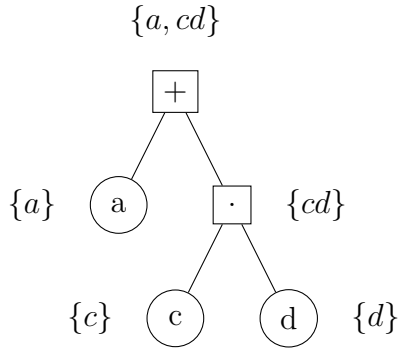

FIGURE 1 – Représentation de l'expression régulière à l'aide d'un arbre syntaxique

Comme on peut voir sur la figure 1 grâce à cette représentation, on peut calculer simplement le langage reconnu par l'expression régulière (ici représenter par les ensemble à côté de chaque arbre).

On comprendra aussi que l'expression $E'(\mathbb{A}) = \epsilon + b^* \cdot a$, dénote le langage $L(E'(\mathbb{A})) = \{\epsilon, b^* \cdot a\} \Leftrightarrow \{\epsilon, a, b^+\}$.

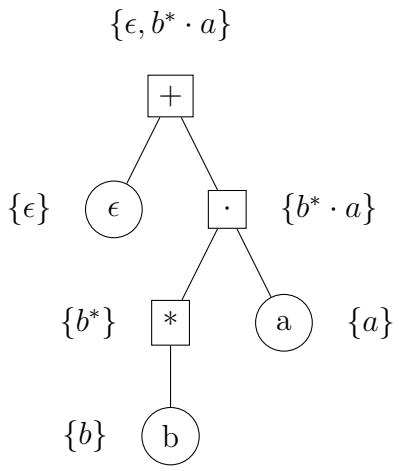

FIGURE 2 – Représentation de l'expression régulière à l'aide d'un arbre syntaxique

2 Fonction sur les *ER*

• Une des fonctions les plus importantes sur les expressions régulières est $flnf$ permettant de calculer $FLNF(E(\Sigma))$ qui est un tuple défini comme ceci :

$$FLNF(E(\Sigma)) = (F, L, \Theta, \delta)$$

- $F \subseteq \Sigma$: Ensemble des premiers symboles de l'expression régulière
- $L \subseteq \Sigma$: Ensemble des derniers symboles de l'expression régulière
- $\Theta = \begin{cases} \epsilon, & \text{si } \epsilon \in L(E(\Sigma)) \\ \varnothing & \text{sinon} \end{cases}$
- $\delta : \Sigma \to S$ avec $S \subseteq \Sigma$.
  Fonction renvoyant les symboles suivant du symbole passer argument.

La fonction $flnf$ a donc comme signature :

$$flnf : E(\Sigma) \to FLNF(E(\Sigma)) \tag{12}$$

Et peut-être calculée de cette manière :

$$flnf(\epsilon) = (\varnothing, \varnothing, \epsilon, \delta) \mid \delta(a) = \varnothing, a \in \Sigma \tag{13}$$
$$flnf(a) = (\{a\}, \{a\}, \varnothing, \delta) \mid \delta(a) = \varnothing, a \in \Sigma \tag{14}$$

$$flnf(E(\Sigma) + G(\Sigma)) = (F \cup F', L \cup L', \Theta \cup \Theta', \delta'') \text{ avec} \quad (15)$$
$$\delta''(a) = \delta(a) \cup \delta'(a) \mid \forall a \in \Sigma$$
$$(F, L, \Theta, \delta) = flnf(E(\Sigma)) \wedge (F', L', \Theta', \delta') = flnf(G(\Sigma))$$

$$flnf(E(\Sigma) \cdot G(\Sigma)) = (F'', L'', \Theta \cap \Theta', \delta'') \text{ avec} \quad (16)$$
$$F'' = F \cup F' \cdot \Theta$$
$$L'' = L' \cup L \cdot \Theta'$$
$$\delta''(a) = \begin{cases} \delta(a) \cup \delta'(a) \cup F', & \text{si } a \in L \\ \delta(a) \cup \delta'(a) & \text{sinon} \end{cases} \mid \forall a \in \Sigma$$
$$(F, L, \Theta, \delta) = flnf(E(\Sigma)) \wedge (F', L', \Theta', \delta') = flnf(G(\Sigma))$$

$$flnf(E(\Sigma)^*) = (F, L, \epsilon, \delta') \text{ avec} \quad (17)$$
$$\delta'(a) = \begin{cases} \delta(a) \cup F, & \text{si } a \in L \\ \delta(a) & \text{sinon} \end{cases} \mid \forall a \in \Sigma$$
$$(F, L, \Theta, \delta) = flnf(E(\Sigma))$$

Exemple :

Prenons par exemple l'expression régulière suivante $E(\mathbb{A}) = a \cdot b + c \cdot d$, avec $\mathbb{A} = \{a, b, c, d\}$. Toujours à l'aide d'un arbre syntaxique, on peut calculer ce que $flnf(E(\mathbb{A}))$ donnerait.

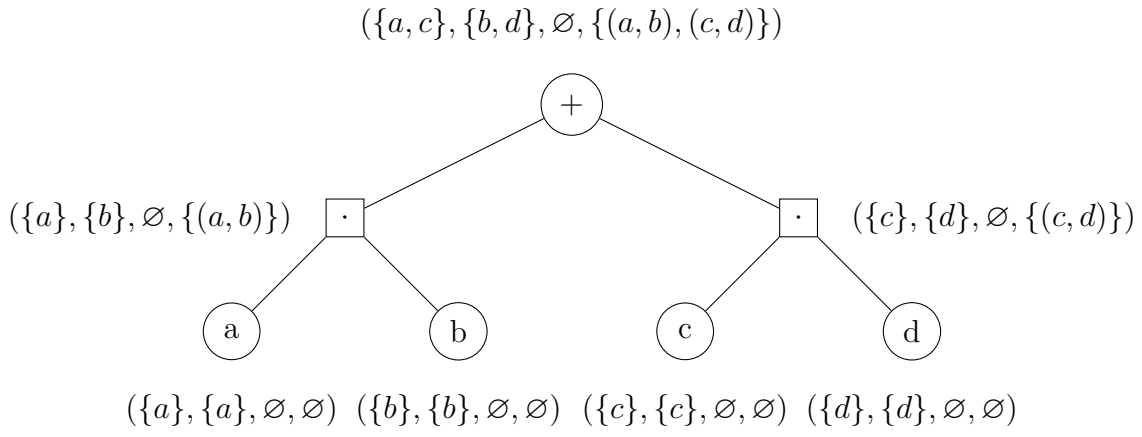

FIGURE 3 – Représentation de l'expression régulière à l'aide d'un arbre syntaxique. Pour des raisons de compréhension δ est représenté à l'aide d'un ensemble de couple.

Il advient que $flnf(E(\mathbb{A})) = \{\{a, c\}, \{b, d\}, \varnothing, \delta\}$ avec δ qui est défini comme ceci :

$$\begin{aligned}\delta(a) &= \{b\}\\ \delta(b) &= \varnothing\\ \delta(c) &= \{d\}\\ \delta(d) &= \varnothing\end{aligned}$$

Un autre exemple pourrait être $E'(\mathbb{A}) = (a + b) \cdot c^*$, avec cet exemple, on voit l'utilité de la parenthèse, car sans elle la concaténation aurait été sur $b \cdot c^*$ et comme dit précédemment (1) son calcul reste le même que si c'était une union.

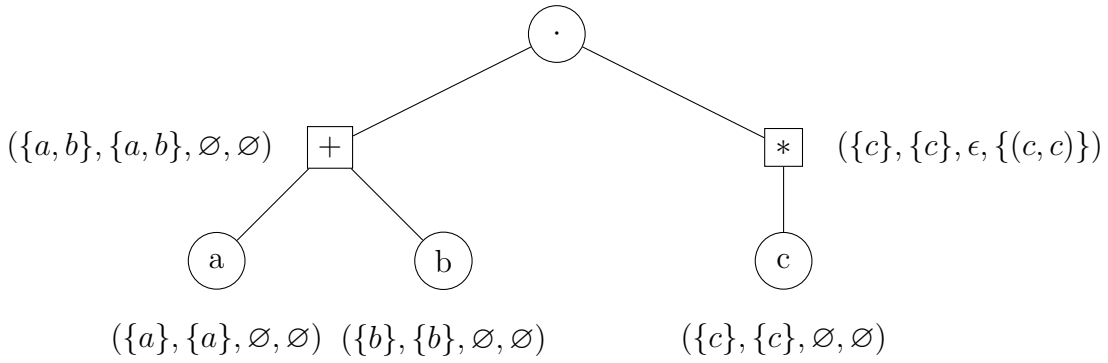

FIGURE 4 – Représentation de l'expression régulière à l'aide d'un arbre syntaxique. Pour des raisons de compréhension δ est représenté à l'aide d'un ensemble de couple.

Ce qui fait que $flnf(E'(\mathbb{A})) = (\{a, b\}, \{a, b, c\}, \varnothing, \delta')$ avec δ' qui est défini comme décrit après :

$$\begin{aligned}\delta(a) &= \{c\}\\ \delta(b) &= \{c\}\\ \delta(c) &= \{c\}\\ \delta(d) &= \varnothing\end{aligned}$$

• Une autre fonction qui s'applique sur les expressions régulières est *linearization*, elle peut paraitre inutile, mais elle nous servira dans la Section III. Sa signature est :

$$linearization : E(\Sigma) \times \mathbb{N} \to (E(\Sigma^{\mathbb{N}}), \mathbb{N})$$
$$\text{avec } \Sigma^{\mathbb{N}} = (\Sigma, \mathbb{N}) \wedge \forall((a, b), (c, d)) \in (\Sigma^{\mathbb{N}})^2 \mid b = d \Rightarrow a = c \Rightarrow (a, b) = (c, d)$$

Elle peut être définie récursivement de cette manière :

$$linearization(\epsilon, n) = (\epsilon, n) \tag{18}$$

$$linearization(a, n) = ((a, n), n + 1) \tag{19}$$

$$\begin{aligned} linearization(E(\Sigma) + F(\Sigma), n) &= (E' + F', n'') \quad \text{avec} \\ (E', n') &\leftarrow linearization(E(\Sigma), n) \\ (F', n'') &\leftarrow linearization(F(\Sigma), n') \end{aligned} \tag{20}$$

$$\begin{aligned} linearization(E(\Sigma) \cdot F(\Sigma), n) &= (E' \cdot F', n'') \quad \text{avec} \\ (E', n') &\leftarrow linearization(E(\Sigma), n) \\ (F', n'') &\leftarrow linearization(F(\Sigma), n') \end{aligned} \tag{21}$$

$$\begin{aligned} linearization(E(\Sigma)^*, n) &= (E'^*, n') \quad \text{avec} \\ (E', n') &\leftarrow linearization(E(\Sigma), n) \end{aligned} \tag{22}$$

Avec cette définition, on peut voir que tous les deuxièmes éléments du couple des symboles du résultat E' avec $linearization(E(\Sigma), n) = (E', m)$, seront supérieurs ou égaux à n et inférieur strict à m.

Exemple :

Si on reprend cette expression régulière $E(\mathbb{A}) = \epsilon + b^* \cdot a$, avec $\mathbb{A} = \{a, b, c, d\}$.

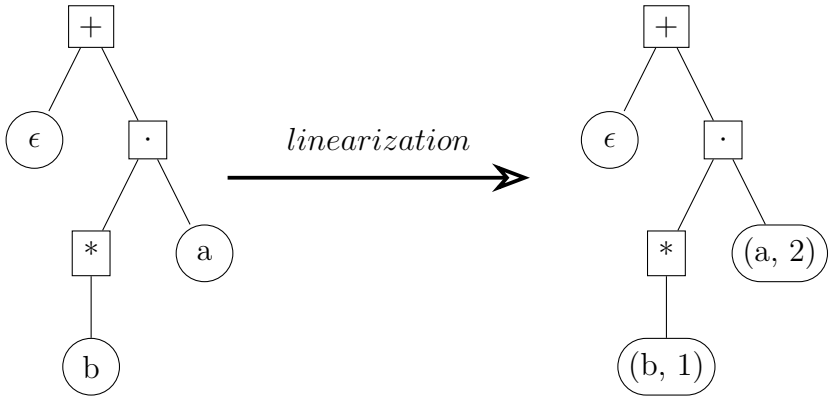

FIGURE 5 – Représentation à l'aide d'un arbre syntaxique de l'expression régulière une fois après avoir fait appel à *linearization* sur elle.

3 Conclusion

On saisit aisément que ces expressions ont beau être simple (peu d'opération comparé aux expressions régulières d'*UNIX*). Elles n'en sont rien pas complètes, on peut voir qu'elles permettent de décrire des langages très complexes et en quantité infinie. Malheureusement, il est difficile de savoir si un mot est reconnu par une expression

régulière simplement. Par exemple est-ce-que le mot *eipipipipipip* est reconnu par cette expression $((((o \cdot \epsilon) + (\epsilon \cdot e)) + ((g \cdot \epsilon) \cdot \epsilon^*)) \cdot ((\epsilon \cdot i) \cdot (p + \epsilon))^*)$? La réponse est oui. C'est pour cela qu'il serait peut-être intéressant d'utiliser une autre structure de donnée pour reconnaitre des mots, comme les automates que nous allons voir maintenant.

II Les automates

Dans cette partie, nous parlerons des automates et plus particulièrement, nous allons parler des automates sans ϵ transition. Pour autant les automates que nous verrons sont tout aussi bons que ceux avec ϵ transition.

1 Définition

Comme dit précédemment un automate est un objet mathématique reconnaissant un langage, on notera $M(\Sigma, \eta)$ l'automate qui a pour transition des valeurs dans Σ et des valeurs 'd'état' dans η. On écrira donc $L(M(\Sigma, \eta))$ pour désigner le langage qu'il reconnait. Un automate est tuple qu'on peut écrire de cette forme $M(\Sigma, \eta) = (Q, I, F, \delta)$ avec :

$$Q \subseteq \eta \quad \text{L'ensemble des états qui constitue l'automate} \tag{23}$$
$$I \subseteq Q \quad \text{L'ensemble des états initiaux} \tag{24}$$
$$F \subseteq Q \quad \text{L'ensemble des états finaux} \tag{25}$$
$$\delta : Q \times \Sigma \to V \text{ avec } V \subseteq Q \quad \text{La fonction de transition} \tag{26}$$

Un automate peut se représenter à l'aide d'un graphe particulier, Par exemple si on veut représenter $M(\Sigma, \eta) = (\{q_1, q_2, q_3, q_4, q_5\}, \{q_1\}, \{q_2, q_3\}, \delta)$ avec $\Sigma = \{0, 1\}$, $\eta = \{q_1, q_2, q_3, q4, q_5\}$ et δ défini comme ceci :

$$\begin{aligned}
\delta(q_1, 0) &= \{q_2\} \\
\delta(q_1, 1) &= \{q_4\} \\
\delta(q_2, 0) &= \varnothing \\
\delta(q_2, 1) &= \varnothing \\
\delta(q_3, 0) &= \{q_3\} \\
\delta(q_3, 1) &= \{q_4\} \\
\delta(q_4, 0) &= \{q_5\} \\
\delta(q_4, 1) &= \varnothing \\
\delta(q_5, 0) &= \{q_4\} \\
\delta(q_5, 0) &= \{q_5\}
\end{aligned}$$

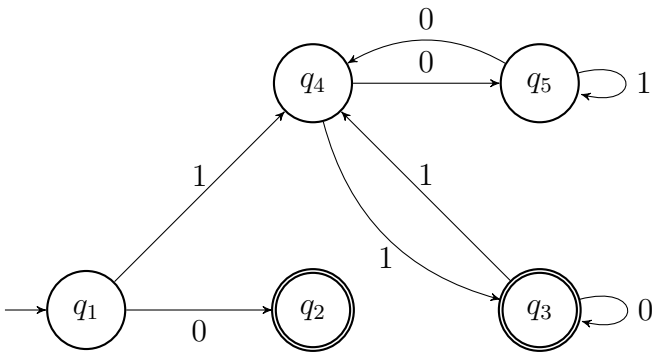

FIGURE 6 – Exemple de représentation graphique d'un automate.

Dans la figure 6, on peut voir que les états initiaux (ici que q_1) ont une petite flèche qui pointe sur lui et que les états finaux ont un double contour. Et que les transitions sont symbolisées par des flèches entre les états et que ces flèches sont labellisées.

On peut étendre la fonction δ de sorte qu'elle ait comme signature :

$$\delta_w : Q \times \Sigma^* \to V \quad \text{avec } V \subseteq Q$$

En la définissant récursivement de telle manière :

$$\delta_w(q, \epsilon) = \{q\} \tag{27}$$

$$\delta_w(q, a.w) = \bigcup_{q'}^{\delta(q,a)} \delta_w(q', w) \quad \text{avec } a \in \Sigma \tag{28}$$

Exemple :

Voici donc quelques exemples de ce que ça nous donnerait si on reprend l'automate utilisé pour la représentation graphique :

$$\delta_w(q_1, 00) = \varnothing \tag{29}$$

$$\delta_w(q_1, 11) = \{q_3\} \tag{30}$$

$$\delta_w(q_1, \varnothing) = \{q_1\} \tag{31}$$

$$\delta_w(q_1, 10111\ldots) = \{q_5\} \tag{32}$$

Propriétés :

Un automate est dit *standard* quand, il ne possède qu'un seul état initial non ré-entrant, aussi défini comme ceci :

$$M(\Sigma, \eta) = (Q, \{i\}, F, \delta) \quad \text{avec} \\ \forall p \in Q, \forall a \in \Sigma \mid i \notin \delta(p, a) \tag{33}$$

Un automate est dit *homogène* quand pour tous les états les transition allant vers cet état ont même valeur. En d'autre terme quand il respecte cette propriété :

$$M(\Sigma, \eta) = (Q, I, F, \delta) \quad \text{avec} \\ \forall (p, q) \in Q^2, \exists! a \in \Sigma \mid q \in \delta(p, a) \tag{34}$$

Donc l'automate représenté sur la figure 6 est standard et non homogène. Car il possède bien un unique état initial (q_1), mais malheureusement q_3, q_4 et q_5 ne respecte pas la propriété pour être homogène, parce qu'ils ont des transitions allant vers eux avec des valeurs différentes.

Les automates pouvant être représentés à l'aide de graphe, on peut étendre les propriétés sur les graphes aux automates. Par exemple, on pourra parler des composantes fortement connexes d'un automate. Autrement dit en partant de n'importe quel état, on peut arriver à tous les autres états. Ainsi ça veut dire que l'automate vérifierait ceci :

$$M(\Sigma, \eta) = (Q, I, F, \delta) \quad \text{avec} \\ \forall (p, q) \in Q^2, \exists w \in \Sigma^* \mid q \in \delta_w(p, w) \tag{35}$$

2 Fonction sur les automates

Une des fonctions la plus importante sur les automates est *accept* qui test si le mot est reconnu par l'automate. C'est-à-dire que si on prend le chemin décrit par le mot donner en argument, on arrive sur un ou plusieurs états finaux. Elle a alors pour signature :

$$accept : M(\Sigma, \eta) \times \Sigma^* \to \mathbb{B}$$

Elle peut être définie simplement comme ceci :

$$accept(M(\Sigma, \eta), w) = \exists q \in \bigcup_{p}^{I} \delta_w(p, w) \mid q \in F \tag{36}$$

3 Conclusion

Comme nous venons de voir les automates sont des outils puissants pour reconnaitre des mots d'un langage. L'un de leur plus grande force est leur simplicité, toutes les opérations sur les automates peuvent donc être automatisé. Ce qui fait que cet objet est très intéressant dans le monde de l'*informatique*. En revanche l'un de ses points faibles est sa représentation, il est difficile de représenter des très gros automates contrairement aux expressions régulières. Il serait alors intéressant de pouvoir convertir une expression régulière en automate. C'est ce que nous allons voir dans la prochaine section.

III Les automates de Glushkov

Le terme 'automates de Glushkov' est un abus de langage, faisant référence aux automates que l'algorithme de transformation d'expression régulière en automate, appelé algorithme de Glushkov produit. Son nom vient de l'informaticien soviétique *Victor Glushkov* qui est son créateur.

1 Définition

Nous appliquerons cet algorithme à l'aide la fonction *glushkov* qui a donc pour signature :

$$glushkov : E(\Sigma) \to M(\Sigma, \mathbb{N})$$

Et a ainsi, on peut définir cette fonction de cette façon :

$$\begin{aligned} glushkov(E(\Sigma)) = (Q, \{0\}, F, \delta) \quad & \text{avec} \\ Q &\leftarrow \{n \mid n \in \mathbb{N} \wedge 0 \leq n < m\} \\ F &\leftarrow \begin{cases} \{n \mid (a,n) \in Last\} \cup \{0\}, & \text{si } Null = \epsilon \\ \{n \mid (a,n) \in Last\} & \text{sinon} \end{cases} \\ \forall q \in \delta(p,a) \mid & \begin{cases} (a,q) \in First, & \text{si } p = 0 \\ (a,q) \in Follow((b,p)) & \text{sinon} \end{cases} \\ (E', m) &\leftarrow linearization(E(\Sigma), 1) \\ (First, Last, Null, Follow) &\leftarrow flnl(E') \end{aligned} \tag{37}$$

Exemple :

Vu qu'un dessin vaut toujours mieux que mile mots, voici un exemple de l'automate résultant de la transformation de cette expression $E(\mathbb{A}) = (a+b) \cdot a^* \cdot b^* \cdot (a+b)^*$.

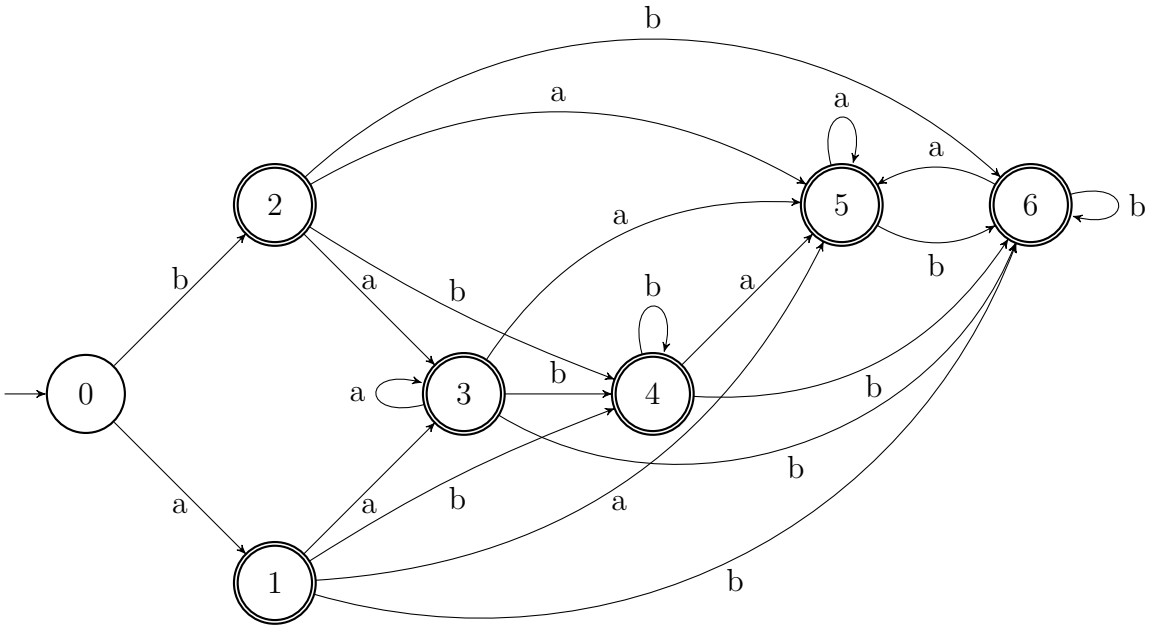

FIGURE 7 – Exemple de représentation graphique de l'automate résultant de $glushkov(E(\mathbb{A}))$.

Comme on peut voir sur la figure 7 les automates produits sont bien souvent gros et peuvent être dur à comprendre, mais une machine peut gérer ça très simplement. Outre sa taille, on peut remarquer qu'il y a des propriétés intéressant sur cette automate. C'est ce que l'on va étudier maintenant.

Propriétés :